# Using Bayesian networks for Candidate Generation in Consistency-based Diagnosis

**Sriram Narasimhan & Ole Mengshoel**
UC Santa Cruz & USRA RIACS
M/S 269-3, NASA Ames Research Center,
Moffett Field, CA 94043
Sriram.narasimhan-1,ole.j.mengshoel@nasa.gov

## Abstract

Consistency-based diagnosis relies heavily on the assumption that discrepancies between model predictions and sensor observations can be detected accurately. When sources of uncertainty like sensor noise and model abstraction exist robust schemes have to be designed to make a binary decision on whether predictions are consistent with observations. This risks the occurrence of false alarms and missed alarms when an erroneous decision is made. Moreover when multiple sensors (with differing sensing properties) are available the degree of match between predictions and observations can be used to guide the search for fault candidates. In this paper we propose a novel approach to handle this problem using Bayesian networks. In the consistency-based diagnosis formulation, automatically generated Bayesian networks are used to encode a probabilistic measure of fit between predictions and observations. A Bayesian network inference algorithm is used to compute most probable fault candidates.

## 1 Introduction

Consistency-based diagnosis techniques [Hamscher, *et al*., 1992] compare model predictions against sensor observations for isolating faults. Structural and behavioral models are used to predict system behavior under hypothesized nominal and faulty conditions. The hypotheses whose predictions best match the sensor observations are reported as the diagnosis. Rather than look at a pre-enumerated set of hypotheses, these approaches use techniques like conflict directed search and backtracking to maintain a short list of consistent hypotheses. This has been applied to discrete [De Kleer and Williams, 1987; Williams and Nayak, 1996; Kurien and Nayak, 2000], discrete-event [Sampath et al., 1996], continuous [Gertler, 1988; Mosterman and Biswas, 1999] and hybrid [Narasimhan and Biswas, 2003] domains.

One of the assumptions in these approaches is that we can say with absolute certainty if predictions and observations are consistent with each other. For example, the Livingstone 2 system [Kurien and Nayak, 2000] uses monitors to make this decision and the TRANSCEND system [Mosterman and Biswas, 1999] uses a symbol generation scheme to make a decision that is statistically robust. If any inconsistencies are detected, information about the inconsistencies (including variables involved in the inconsistencies and possibly the direction or magnitude of the inconsistencies) is used to guide the search for alternate fault candidates. Fault candidates that resolve the observed discrepancies (resolving conflicts entailed by the discrepancies for example) are generated according to some user-defined criteria (typically based prior probabilities and size of the chosen candidates.

However in a lot of situations it may be very difficult to determine if model predictions exactly match sensor observations. Some reasons for this difficulty include (i) imperfect/abstract models resulting in imprecise predictions, (ii) sensor noise resulting in imprecise observations, and (iii) uncertain operating conditions and environment resulting in imprecision predictions and observations. An error in making this binary decision (either reporting a discrepancy when there is none or not reporting a discrepancy when one exists) will result in erroneous diagnosis results. Additionally the actual degree of fit between predictions and observations for individual variables might provide useful diagnostic information allowing us to limit the search for fault candidates (resulting in faster diagnosis).

Some probabilistic approaches address this problem by setting up a Bayesian formulation (as opposed to the consistency-based approach) to solve problem [Dearden and Clancy, 2002; Hofbaur and Williams, 2004;]. Typically candidates have weights/probabilities associated with them and these weights are updated at each time step based on the model and observed values. Candidates that represent faults may be introduced in several ways including importance sampling [Dearden and Clancy, 2002] and using a consistency-based diagnosis scheme [Narasimhan, *et al*., 2004].

In this paper we propose an alternate approach using Bayesian networks, which attempts to solve this problem within a consistency-based framework but using a Bayesian network as a component. In our approach, a consistency-based diagnosis engine is the driving force. The engine is responsible for maintaining a set of candidates "consistent" with the observations seen so far. At each time step the engine tests each candidate for consistency with the current observations. Rather than looking for a binary decision on the consistency of the candidate, the consistency test is used

to provide probabilistic measure of the degree of fit between predictions and observations for each observed variable. The engine then utilizes a *Bayesian network* (BN) that encodes the structure associated with the current model as well as probabilistic information in the form of prior probabilities of faults and the probability of fit for observable variables (computed in the consistent testing step earlier). The BN can then be queried for the most probable assignment of values to all variables, a subset of which correspond to faults in the system. This information can then be used to update the candidate set maintained by the diagnosis engine.

In this paper we will focus on a specific consistency-based diagnosis system called HyDE [Narasimhan and Brownston, 2007] and show our approach works in that framework. However the ideas are general and can be adapted to other consistency-based diagnosis systems. We will describe the makeup of the BN, how it can be constructed automatically from existing models in the HyDE framework (this is the only part that would be different for a different diagnosis technology) and how it can be integrated with a consistency-based diagnosis engine. Initial experiments, with a two tank system, show a significant improvement in the diagnostic accuracy, when our novel approach is used.

The rest of paper is divided as follows. Section 2 presents some back ground on BN and the consistency-based diagnosis paradigm we will be assuming. Section 3 presents the Hybrid Diagnosis Engine (HyDE) and its diagnosis architecture. Section 4 presents our novel approach of using BN for candidate generation. Section 5 presents some examples and results from using this combined approach. Section 6 presents conclusions and scope for future work.

# 2 Background

## 2.1 Consistency-based Diagnosis

Several interpretations of consistency-based diagnosis exist in the literature [Hamscher, *et al*., 1992]. In order to take into account the hybrid and dynamic nature of the systems being diagnosed, we will be using the following representation of consistency-based diagnosis as our basis. We assume that the consistency-based diagnosis uses a "generate and test" paradigm to detect and isolate faults. The diagnosis engine maintains a set of *consistent candidates* which is updated at each time step by adding or pruning candidates based on the observations from sensors. The candidates represent hypotheses about faults that have occurred in the system with associated time stamps. At each time step candidates in the candidate set are tested for consistency against observations available at that time step. If a candidate is found to be inconsistent, it is dropped from the set. New candidates are generated by backtracking in the model from the point of inconsistency (typically an intermediate step of generating conflicts is used). The newly generated candidates can be tested and added to the candidate set if found to be consistent with observations. The test for consistency uses models that can be used to predict what the system is expected to do. The predictions can be compared against observations to check for consistency.

## 2.2 Bayesian networks

A *Bayesian network* (referred to as BN in the rest of this paper), or a belief network, is a probabilistic graphical model that represents a set of variables and their probabilistic independencies. The term "Bayesian networks" was coined by Judea Pearl [Pearl, 1985] to emphasize three aspects:

1. The often subjective nature of the input information.
2. The reliance on Bayes's conditioning as the basis for updating information.
3. The distinction between causal and evidential modes of reasoning, which underscores Thomas Bayes's posthumous paper of 1763.

Bayesian networks are directed acyclic graphs whose nodes represent variables, and whose arcs encode conditional independencies between the variables. Nodes can represent any kind of variable, be it a measured parameter, a latent variable or a hypothesis. If there is an arc from variable $x_1$ to another variable $x_2$, $x_1$ is called a parent of $x_2$, and $x_2$ is a child of $x_1$. Associated with each variable $x_i$ is a joint probability distribution which specifies the probability of $x_i$ taking each value in its domain for all possible value assignments for the parents of $x_i$.

Efficient algorithms exist that perform inference and learning in Bayesian networks. Because a BN is a complete model for the variables and their relationships, it can be used to answer probabilistic queries about them. For example, the BN can be used to find out updated knowledge of the state of a subset of variables when other variables values (called *evidence*) are known. This process of computing the posterior distribution of variables given evidence is called probabilistic inference.

Formally, BN can be defined as BN = ({X},{E},{P}) where $X=\{x_1,x_2,\ldots,x_m\}$ are the m variables in the BN with associated *conditional probability distributions* $P=\{p_1,p_2,\ldots,p_m\}$ and $E=\{e_1,e_2,\ldots,e_n\}$ represent the n arcs between variables in {X} with $e_i = x_j \rightarrow x_k$ for $i \neq k$.

# 3 Hybrid Diagnosis Engine (HyDE)

Hybrid Diagnosis Engine (HyDE) is a model-based reasoning engine for hybrid (discrete + continuous) diagnosis. HyDE is able to diagnose multiple discrete faults using consistency checking between prediction from hybrid models and sensor observations. We first describe the models used by HyDE in its reasoning.

## 3.1 HyDE Models

A HyDE model is made up of the following elements:

1. The set C of the *components* of the system.
2. The set L of operating modes of all components called *Locations*.
3. The set TR of allowed *transitions* between the locations of the same component. Each transition tr $\in$ TR is of the form $l_{from} \rightarrow l_{to}$,g where $l_{from} \in c$ & $l_{to} \in c$ for

some c Є C and g is a *guard* indicating the conditions under which the transition may be taken. An empty guard is used to encode a special kind of transition called *unguarded transition*. Unguarded transitions can be used represent, among other things, *faults* in the system.

4. The set V of *variables* and the set VD of *domains* associated with the variables, specifying the allowed values (data types) for the variables.
5. The *propagation model* PM specifies the behavior of the system within a time step as relations over variables. This includes:
    a. Global model $PM_g = R_g(V)$, where $R_g$ is the global set of relations constraining values of variables. These relations are valid at all times.
    b. Local models $PM_l = R_l(V)$ for each l Є L, where $R_l$ is the set of local relations constraining values of variables. These relations are applicable only when the system (corresponding component) is in location l.
6. The *integration model* IM specifies the evolution of values each variable across time steps. It specifies how state variable values at one time step can be computed from state variable values and derivative (of state) variable values at the previous time step.
7. The *dependency model* DM specifies qualitative how variables in the model are influenced by local relations.

### 3.1 HyDE Reasoning

The reasoning algorithm in HyDE (illustrated in Figure 1) essentially maintains a set of *candidates* D. The goal of HyDE is to find the candidates that best match the observations seen so far. Each candidate contains a possible trajectory of system behavior evolution represented in the form of a *hybrid state* (HS) history and transition history. The hybrid state is a snapshot of the entire system state at any single instant. It associates all components with their current locations and all variables with their current values. The hybrid state history tracks hybrid states at beginning and end of all time steps of the system behavior evolution. The transition history tracks transitions taken by all components at all time steps. In order to avoid monotonic growth in the histories, a user-defined parameter called history window is used to restrict the length of history saved.

At each time step $t_i$ in the reasoning process, HyDE tests each candidate for consistency with observations at $t_i$. This involves computing the hybrid state at the end of $t_i$ ($HS_{ti+}$) from the HS at the beginning of $t_i$ ($HS_{ti-}$) and the model entailed by the HS. A subset of variable values in $HS_{ti+}$ corresponds to predictions for observed variables. These predicted values are compared against corresponding observations. If they are found to be inconsistent then a *candidate generator* is created which finds unguarded transitions (one or more) that can possibly *resolve* the inconsistency. A transition can possibly resolve an inconsistency if the relations from the source location of the transition directly or indirectly influence the variable found to be inconsistent. Since multiple transitions may resolve an inconsistency and typically multiple inconsistencies (across time) may occur after a fault, a search process is needed to identify the most important transitions. Importance is judge based on criteria like maximum prior probability and minimum size. A candidate manager is responsible for pruning candidates that were found to be inconsistent and adding candidates by querying the candidate generators. The reasoning algorithm can be summarized as follows:

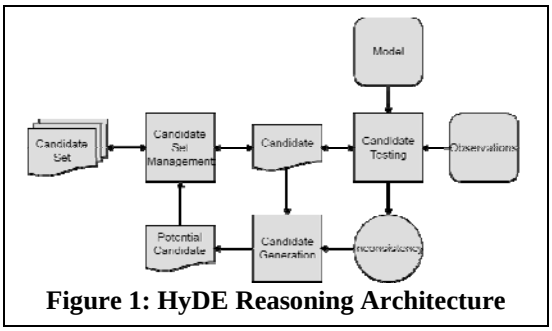

**Figure 1: HyDE Reasoning Architecture**

1. Initialize consistent candidate set with the *empty* candidate $D_c = \{d_{empty}\}$ where $d_{empty} = (HS_0, \{\})$ and $HS_0$ is the initial hybrid state of the system which is assumed to be known and the system is assumed to be in nominal state (no unguarded transitions have been taken).
2. Repeat at each time step $t_i$ for each candidate $d_k$ Є $D_c$
    2.1. Advance the hybrid state from the end of the previous time step ($HS_{ti-1+}$) to the beginning of the current time step ($HS_{ti-}$).
    2.2. Compute HS at end of time step ($HS_{ti+}$) from $HS_{ti-}$, current values of input variables ($U_{ti}$), global constraints ($R_g$) and local constraints associated with current system location obtained from $HS_{ti-}$ ($R_{lti}$).
    2.3. Compare sensor values for observed variables $V_{obs}$ with predicted values $V_{obs}(HS_{ti+})$ to identify inconsistent variables $V_{inconsistent}$ C $V_{obs}$.
    2.4. In the dependency model, trace backwards from each v Є $V_{inconsistent}$ to identify all local relations that influence v. The locations associated with these relations together form a conflict (if the system is assumed to be in these locations then v becomes inconsistent).
    2.5. Possible transitions that can resolve a conflict are the unguarded transitions out of the locations that form the conflict. Compute a set of unguarded transitions TR (selecting the best set based on user-specified ranking criteria) that resolve conflicts associated with all inconsistent variables $V_{inconsistent}$. TR and the HS history from $d_k$ can then be used to generate a new candidate $d_{potential}$. $D_{potential}$ can then be tested to see if it actually resolves the conflict by tracking its behavior prediction. If it is found consistent with the observations then it is added to the candidate set ($D_c = D_c$ U $d_{new}$) else it is discarded.
    2.6. Remove the inconsistent candidate from the candidate set ($D_c = D_c - d_k$).

For more details about the HyDE reasoning algorithm please refer to [Narasimhan and Brownston, 2007].

### 3.2 Bayesian networks for Candidate Generation in HyDE

When we look at the steps of the HyDE reasoning algorithm, step 2.3 assumes that we can make a binary decision on which observations are inconsistent with predictions. Based on this decision, new candidates may be added (and the inconsistent one eliminated) using a conflict directed search. This approach fails when inconsistencies cannot be detected accurately. It also fails to make use of the magnitude/degree of the inconsistency which might be a useful guide when searching for candidates. We propose a modified HyDE reasoning algorithm that uses the degree of inconsistency (rather than expecting to make a binary decision based on it) by constructing a BN and generating candidates to be tested by computing the most probable hypothesis in the BN.

The basic idea is to estimate a probabilistic measure of the (in)consistency between model predictions and sensor observations. Depending on the domain of the variable and sensor noise properties this may be achieved by simply thresholding or by other means like the probability distribution function of a Gaussian distribution (which we will see in the example later). In order to use this probabilistic measure effectively, we use an automatically constructed BN in place of the dependency model for candidate generation. The structure of the BN is determined by the propagation and integration model (Bullets 5 & 6 from the HyDE model description). The conditional probability distributions in the BN are obtained from prior probabilities of unguarded transitions and the estimated probabilistic measured of (in)consistency between predictions and observations.

We modify HyDE reasoning algorithm in the following ways. We add an initialization step (1a) that initializes the BN for the initial candidate.

1a. Initialize the $BN_{empty}$ associated with nominal candidate with nodes for locations of components at start of time step 0. If $l_{ji}$ be the location of component $c_i$ in $HS_0$ then $BN_{empty} = (\{x_{c1},x_{c2},\ldots,x_{cm}\},\{P_{c1},P_{c2},\ldots,P_{cm}\},\{\})$ where $P_{ci} = \{ p(c_i \text{ in } l_{ji}) = 1.0, p(c_i \text{ in } l_{jk} | k \neq i) = 0.0\}$

Steps 2.3 through 2.6 are modified as follows:

2. Repeat at each time step $t_i$ for each candidate $d_k \in D_c$

2.3. Compare sensor values for observed variables $V_{obs}$ with predicted values $V_{obs}(HS_{ti+})$ to assign probabilities to each variable being consistent. For $v \in V_{obs}$, $p(v=\text{CONSISTENT}) = \delta(v_{ti}, \hat{v}_{ti})$ where $\delta$ is a user-customizable comparison function (for example thresholding). $p(v=\text{INCONSISTENT}) = 1 - p(v=\text{CONSISTENT})$.

2.4. Construct the BN for $t_i$ and for the transition from time step $t_{i-1}$ to $t_i$ and append it to BN for the candidate $BN_{dk} = BN_{dk} \cup (\{X_{ti}\},\{E_{ti}\},\{P_{ti}\})$. This step is discussed in more detail in the next section. Truncate parts of the BN referring to time steps prior to the horizon (time steps outside the history window). $BN_{dk} = BN_{dk} - (\{X_{tH}\},\{E_{tH}\},\{P_{tH}\})$ for all $t_H < t_i - t_{history}$ where $t_{history}$ is a user specified time history window.

2.5. Query $BN_{dk}$ for the Most Probable Explanation (MPE) which provides the most likely assignment of values for all variables in the BN. A subset of these variables corresponds to transitions taken by all components at all time steps in the time history window. This subset will be used to generate a new candidate $d_{new}$ with hybrid state obtained from $d_k$. If $d_{new} = d_k$ then nothing needs to be done. However if $d_{new} \neq d_k$ then $d_{new}$ is added to the set of consistent candidate ($D_c = D_c \cup d_{new}$).

2.6. Check the probability of the $d_k$ in the BN and if it falls below a certain threshold $p_{cutoff}$ then remove $d_k$ from $D_c$ ($D_c = D_c - d_k$).

We describe the automatic construction of the BN models (step 3.4) for a specific candidate $d_k$ at a specific time step $t_l$ in section 4.

### 3.3 Example and Results

To illustrate the advantage of using the proposed approach, we present a two tank example. The system consists of two tanks with outlet pipes from both tanks and a connecting pipe between the two tanks. A flow source feeds liquid into the first tank. Each of the pipes and the source can be in nominal mode (resistance is specified constant) or in highResistance mode (resistance is 5 times the specified constant). The actual highResistance fault in the system is usually not exactly 5 times but could vary between 4 and 6 times the resistance. The out flows from both outlet pipes are the only observed variables. The sensors associated with the out flows are assumed to have White Gaussian noise.
The HyDE model of this example is illustrated in Figure 2.

In this example, due to feedback effects, all of the faults will eventually impact both observed variables. However the presence of integrating elements in the form of tank capacitances will introduce a time delay in the propagation of fault effects. For example, if Pipe1 is in highResistance mode then we should see an immediate influence in the observed Pipe1 outflow while the influence on Pipe2 outflow will take a few time steps to manifest. If we use a purely consistency based diagnosis then we have to wait until both observations have deviated (because no fault influences only one observation) and even then all fault candidates are possible and they will be tested in the order of prior probabilities (Pipe12, Pipe2, Pipe1, Source in that order for this example). The presence of sensor noise and uncertainty about actual magnitude makes it very difficult to absolutely de-

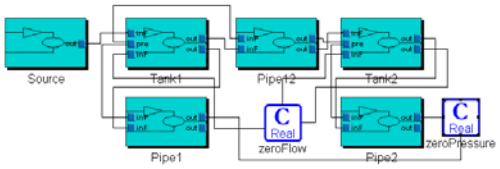


**Figure 2: HyDE model of Two Tank System**

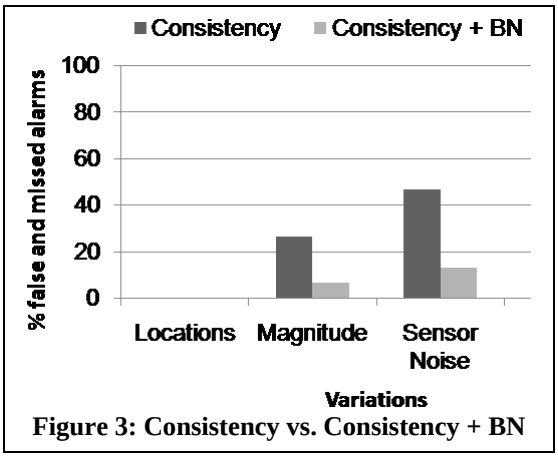

**Figure 3: Consistency vs. Consistency + BN**

termine if sensor observations deviate from model predictions. If the sensor noise is high or if the actual fault magnitude is not close to 5, the chances of misclassification are also quite high.

We ran 34 scenarios by varying the location (4 scenarios) and magnitude of fault (15 scenarios) as well as the sensor noise level (15 scenarios). If a sensor observation was not found to be within 2 standard deviations of a Gaussian distribution (95% of the time values sampled will be within 2 standard deviations) with mean set to the model prediction and standard deviation set based on level of sensor noise then it was considered to be inconsistent. The results are summarized in Figure 3 (Dark Rectangles on the left). We can clearly see that there are a large number of missed alarms and false alarms when using a purely consistency-based approach.

For the second set of experiments we used the proposed modification to HyDE using BN. The BN was automatically generated from HyDE model (one example is illustrated in Figure 4) and probabilities for observed variables were computed using the probability distribution function for Gaussian distribution given by:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

Figure 3 (Light Rectangles on the right) shows the results of integrating the BN in the consistency-based diagnosis framework. There is a significant reduction in both missed and false alarms since this approach does not commit to a decision which may later turn out to be erroneous. However we do pay the penalty of using more time and computational resources for the BN computation.

# 4 Automatic Generation of Bayesian networks

We noted that in step 3.4 of the modified HyDE reasoning algorithm presented in the previous section, the BN for each candidate $d_k$ ($BN_{dk}$) is augmented with the BN fragment at the current time step $t_j$ ($BN_{ti}$) and BN fragment for the transition from previous time step $t_{i-1}$ to current time step $t_i$ ($BN_{ti-1 \to ti}$). $BN_{dk} = BN_{dk} \cup BN_{ti} \cup BN_{ti-1 \to ti}$.

We first describe the automatic generation of $BN_{ti}$ from the HyDE models and then describe the generation of $BN_{ti-1 \to ti}$.

## 4.1 Bayesian network within a time step

In order to generate the BN at a specific time step $t_i$ for candidate $d_k$, we find the system location predicted by $d_k$ at the beginning of $t_i$. This can be obtained from the hybrid state predicted by $d_k$ at the beginning of $t_i$ ($HS_{ti-}$). $SL_{ti} = \{l1_{ti}, l2_{ti}, \ldots, ln_{ti}\} = SL(HS_{ti-})$ where $lj_{ti}$ corresponds to the location of component $c_j$. The next step is to construct the constraint system model ($R_{ti}$) at $t_i$ as predicted by $d_k$. This should already have been constructed in step 3.2 and can be re-used in the generation of $BN_{ti}$.

In the reminder of this section we will omit the subscript $t_i$ for convenience. BN is computed as follows. The nodes {X} in the BN consist of nodes corresponding to variables in the model ($X_v$) and nodes corresponding to components in the model ($X_c$): $X = X_v \cup X_c$

- $X_v = \{x_{v1}, x_{v2}, \ldots, x_{vk}\}$ where $x_{vi} \in \{CONSISTENT, INCONSISTENT\}$ corresponds to variable $v_i$ in the constraint system model.
- $X_c = \{x_{c1}, x_{c2}, \ldots, x_{cm}\}$ where $x_{ci} \in Locations(l_i)$ corresponds to Component $c_i$ in the constraint system model.

The arcs {E} between nodes in X are computed based on the currently valid relations $R = R_g \cup R_{SL}$. We use the following algorithm for generating arcs in the BN:

1. Create two variable lists, KNOWN ($X_{vk}$) & UNKNOWN ($X_{vu}$). Move all input and state variables to $X_{vk}$ and all other variables to $X_{vu}$.
2. Create a TOBEPROCESSED relations list (RT) and move all relations in R to RT.
3. Repeat until $RT \neq \{\}$
   a. Find a relation $r(X_{Vka}, X_{Vua}) = R \in RT$ where $X_{Vka}$ represents variables in r belonging to $X_{vk}$ and $X_{Vua}$ represents variables in r belonging to $X_{vu}$, such that size of $X_{Vu}$ is minimum among all $r \in R$. In other words find the relation with fewest numbers of UNKNOWN variables.
   b. Create a bi-partite graph with nodes from $X_{Vka}$ on one side and nodes from $X_{vua}$ on the other side and arcs from all nodes in $X_{Vka}$ to all nodes in $X_{Vua}$. Make all the UNKNOWN variables in r depend on KNOWN variables in r. This encodes our intuition that the KNOWN variables in r would be used to compute values for the UNKNOWN variables in r.
   c. If r is a local constraint (i.e., $r \in R_{SL}$) belonging to location $l_m$ of component $c_m$ then add an arc from the $x_{cm}$ to all $x \in X_{Vua}$. This encodes the dependency that a local relation will only be used when the system is in that location.
   d. Move all variables in $X_{Vua}$ from $X_{vu}$ to $X_{vk}$ i.e., For each $x_{Vui} \in X_{Vui}$, $X_{Vui} = X_{Vui} - v_{ui}$ & $X_{Vki} = X_{Vki} \cup x_{Vui}$. All UNKNOWN

variables in r can now considered to be KNOWN through computation.

e. Remove r from the TOBEPROCESSED list. R = R - r.

The conditional probability tables {P} are computed as follows:

1. For all $x \in X_v$ such that x corresponds to an input or state variable set p(x=CONSISTENT) = 1.0, p(x=INCONSISTENT) = 0.0. We will see in the next section that if there are arcs to the state variables from the BN fragment at previous time step then the P for the state variables will be different.
2. For all other variables (non-input and non-state) $x \in X_v$ that have incoming arcs only from variables in $X_v$ P is set to
   a. p(x=CONSISTENT|for all $x_v \in X_v$, $x_v$ = CONSISTENT) = 1.0
   b. p(x=INCONSISTENT|for all $x_v \in X_v$, $x_v$ = CONSISTENT) = 0.0
   c. p(x=CONSISTENT|there exists $x_v \in X_v$, $x_v$ = INCONSISTENT) = 0.0
   d. p(x=INCONSISTENT| there exists $x_v \in X_v$, $x_v$ = INCONSISTENT) = 1.0
3. For variables $x \in X_v$ with arcs from other variables $X_v$ as well as arcs from $x_c$(For any variable $x \in X_v$ there can only be one variable $x_c \in X_c$ with an arc to x) the P when $x_c = l_{current}$ where $l_{current}$ is current location of component c in SL is set 1 and 0 other wise
   a. $p(x=CONSISTENT|x_{ca} \neq l_a) = 0.0$
   b. $p(x=INCONSISTENT|x_{ca} = l_a) = 1.0$.
4. For all variables $x_c \in X_c$, we set the probability of the variable being in the location predicted by the candidate to be 1 and probability of any other location to be 0
   a. $p(x_{ca} = l_{current}) = 1.0$
   b. $p(x_{ca} = l_b \text{ for all } l_b \neq l_{current}) = 0.0$.

   As we will see in the next section, if $x_c$ has an arc from $x_c$ at the previous time step then the P is computed differently.

## 4.2 Bayesian network across time steps

Once we have generated the BN fragment for candidate $d_k$ at a specific time step $t_i$ ($BN_{ti}$) we have to connect it to $BN_{dk}$ (associated with earlier time steps). Let $BN_{ti-1} = (\{X_{ti-1}\},\{E_{ti-1}\},\{P_{ti-1}\})$ represent the part of $BN_{dk}$ corresponding to the previous time step. We augment $BN_{dk}$ as follows:

- First we add a new set of variables $X_T = \{x_{T1},x_{T2},\ldots,x_{Tn}\}$ where $x_{Ti}$ corresponds to an unguarded transition taken by component $c_i$. Hence there will be n such variables where n is the number of components. The domain for any one of these variables $x_{Ta} \in X_T$ will be all the unguarded transitions out of $l_{current}$ where $l_{current}$ is the current location of component $c_a$ plus an additional transitional called the self transition ($T_{self}$) which if taken keeps the component in the same location.
- Next we add arcs $E_T$ that represents the conditional dependence of the location of component at $t_i$ on the transition the component takes between $t_{i-1}$ and $t_i$. Each arc $e_T$ is of the form $x_T \rightarrow x_c[t_i]$.
- Then we add arcs $E_c$ indicating the conditional dependence of the location of a component at time step $t_i$ on the location of the same component at time step $t_{i-1}$. Each arc $e_c$ is of the form $x_c[t_{i-1}] \rightarrow x_c[t_i]$.

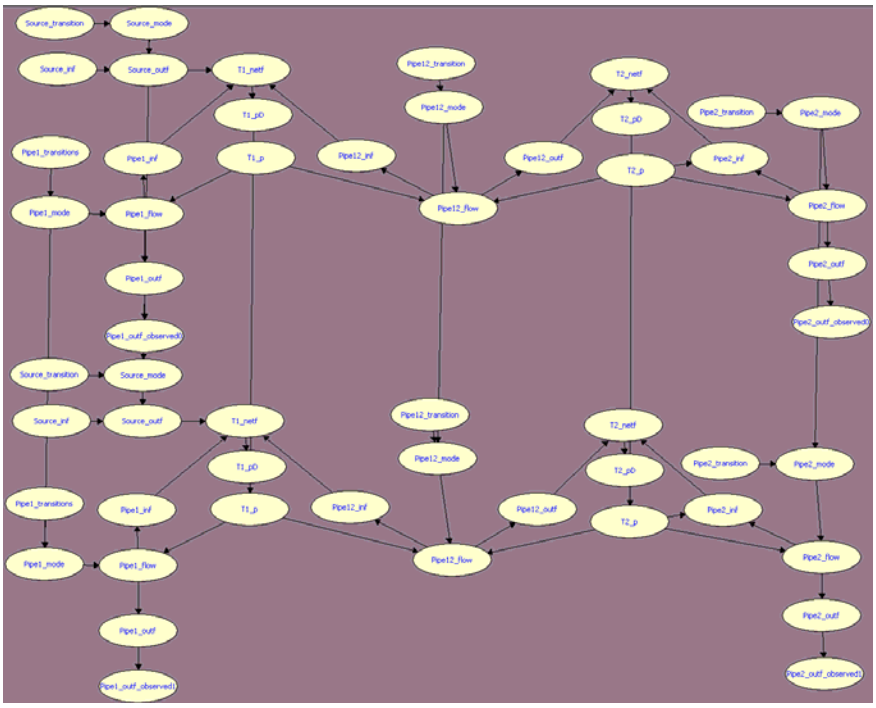


**Figure 4: Bayesian network for Two Tank System**

- Then we add arcs $E_v$ to represent the conditional dependence of state variables on derivative variables as determined by the Integration model. For each state variable $x_v$, we determine the set of derivative variable $X_d$ that will be required to compute the value of $x_v$. We then draw arcs from each $x_d \in X_d$ to $x_v$.
- The P for $x_c[t_{i-1}]$ is unchanged since it has no new incoming arcs.
- The P for $x \in X_{Ti}$ is set to the prior probabilities of the corresponding unguarded transitions.
- The P for $x_c[t_i]$ associated with component c is computed as follows. Based on the location of c at $t_{i-1}$ and the transition T taken by component between $t_{i-1}$ and $t_i$, we can deduce the location of c at $t_i$. The probability for this location is set to 1 and the probabilities for the rest of the locations are set to 0.
  - $p(x_{ca}[t_i]=l_b|x_{ca}[t_{i-1}]=l_c \ \& \ x_T=l_c \rightarrow l_b) = 1.0$
  - $p(x_{ca}[t_i]=l_b|x_{ca}[t_{i-1}]=l_c \ \& \ x_T=l_c \rightarrow l_d, \ l_d \neq l_b) = 0.0$
- The P for $x_v[t_i]$ for all state variables is updated by setting it to be CONSISTENT when all derivative variables that influence it from the previous time step ($X_d[t_{i-1}]$) and corresponding state variable from previous time step ($x_v[t_{i-1}]$) are CONSISTENT
  - $p(x_v[t_i]=\text{CONSISTENT}|\ X_d[t_{i-1}]=\text{CONSISTENT} \ \& \ x_v[t_{i-1}]=\text{CONSISTENT}) = 1.0$
  - $p(x_v[t_i]=\text{CONSISTENT}|$ there exists $x_d \in$ Xd such that $x_d[t_{i-1}]=\text{INCONSISTENT} \ | \ x_v[t_{i-1}]=\text{INCONSISTENT}) = 0.0$
  - $p(x_v[t_i]=\text{INCONSISTENT}) = 1 - p(x_v[t_i]=\text{CONSISTENT})$

## 6 Conclusion and Future Work

We presented a new approach for incorporating probabilistic information in a consistency-based diagnosis framework. This approach uses automatically constructed BN models for candidate generation. The conditional probability distributions in the BN are based on probabilistic measures of the consistency between model predictions and observations. BN inference can be used to identify the most likely hypothesis which if different from the original candidate is used to generate a new candidate.

The approach presented in this section differs from typical consistency-based approaches in that it is able to deal with uncertainty when comparing observations and predictions from the model. It differs from typical BN approaches to diagnosis in that it uses BN inference only for the candidate generation rather than entire diagnosis process [Roychoudary et al., 2006].

This new approach combining BN and consistency-based is meant to be useful when a significant amount of uncertainty exists in testing candidates for consistency. In such situations the use of the proposed approach will result in fewer false alarms and missed alarms. Additionally because fewer candidates are likely to be tested, there is an improvement in time and memory performance as well. However in some cases there might be an increase in time and memory performance because of the need to construct the BN and perform inference on it. For the purposes of this paper we used the SAMIAM tool from UCLA (http://reasoning.cs.ucla.edu/samiam/) for Bayesian network inference. In future work we would like to explore the use of strategies to improve the performance of the BN inference including pre-generation and compilation approaches suggested by Darwiche [Chavira and Darwiche, 2007].

Once we have established the framework to integrate BN inference for candidate generation in consistency-based diagnosis there is scope for incorporating several kinds of uncertainty in the reasoning. For example, sensor noise can be modeled directly by the conditional probability distributions for the BN variables corresponding to observable variables. Similarly it is possible to incorporate uncertainty about model parameters in the BN. In future work we would like to explore the extension of BN to support other such forms of uncertainty and see how that improves the sensitivity of the diagnosis algorithm.

As we mentioned earlier this approach can be easily adapted to work with other consistency-based techniques as well. This can be achieved by providing an algorithm to construct the BN from whatever modeling paradigm is being used. For example, the temporal causal graphs in the TRANSCEND system [Mosterman and Biswas, 1999] already provide a structure that can be used to construct the BN. Even more diagnostic power is possibly by changing the BN variable states to (-, 0, +) as used in the TRANSCEND reasoning algorithms. The signal to symbol transformation algorithms have to be modified to output a probabilistic measure of consistency (or a probability distribution over -, 0, + if those states are used). Using this approach, TRANSCEND would not be forced to commit to a set of candidates generated by the initial backtracking. Alternately it might be possible to start the fault isolation reasoning earlier since it is not essential for 100% accuracy in determining symbols.

## Acknowledgments

We would like to thank the HyDE team members Lee Brownston and David Hall for providing useful feedback on the approach discussed in this paper. We would also like to thank Adnan Darwiche's group at UCLA for use of the SAMIAM tool.

## References

[Hamscher, *et al*., 1992] Walter Hamscher, Luca Console, and Johan De Kleer. *Readings in Model-based Diagnosis.* San Mateo, CA: Morgan Kaufmann, 1992.

[De Kleer and Williams, 1987] Johan De Kleer and Brian C. Williams. *Diagnosing multiple faults,* Artificial Intelligence, 32(1):97–130, 1987.

[Sampath, et al., 1996] Sampath, M.; Sengupta, R.; Lafortune, S.; Sinnamohideen, K.; Teneketzis, D.C. Failure Diagnosis using Discrete-Event Models. *IEEE Transactions on Control Systems Technology*, 1996. **4**: pp. 105-124.

[Williams and Nayak, 1996] Brian Williams and Pandu Nayak. *A model-based approach to reactive self-configuring systems*, in AAAI, pp. 971–978, (1996).

[Kurien and Nayak, 2000] James Kurien and Pandu Nayak. *Back to the Future with Consistency-based Trajectory Tracking*, AAA/IAAI 2000, pp370-377.

[Gertler, 1988] Janos Gertler. *Fault Detection and Diagnosis in Engineering Systems.* New York: Marcel Dekker, 1988.

[Mosterman and Biswas, 1999] Pieter J. Mosterman and Gautam Biswas. *Diagnosis of continuous valued systems in transient operating regions.* IEEE Transactions on Systems, Man, and Cybernetics, 1(6):554–565, 1999.

[Dearden and Clancy, 2002] Richard Dearden and Dan Clancy. *Particle Filters for Real-time Fault Detection in Planetary Rovers*, in Proc. 13th International Workshop on Principles of Diagnosis (DX '02), Semmering, Austria, pp. 1-6, 2002.

[Hofbaur and Williams, 2004] Michael Hofbaur and Brian C. Williams. *Hybrid Estimation of Complex Systems*, IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics, 2004, Special Issue on Diagnosis in Complex Systems: Bridging the methodologies of the FDI and DX Communities, 2004, pp. 2178-2191.

[Narasimhan, *et al*., 2004] Sriram Narasimhan, Richard Dearden, and Emmanuel Benazera. *Combining Particle Filters and Consistency-based Approaches for Monitoring and Diagnosis of Stochastic Hybrid Systems*, 15th International Workshop on Principles of Diagnosis (DX04), Carcassonne, France, June 2004.

[Narasimhan and Brownston, 2007] Sriram Narasimhan and Lee Brownston. *HyDE – A General Framework for Stochastic and Hybrid Model-based Diagnosis*, in Proc. 18th International Workshop on Principles of Diagnosis (DX '07), Nashville, USA, pp. 162-169, 2007.

[Pearl, 1985] Judea Pearl (1985). *Bayesian networks: A Model of Self-Activated Memory for Evidential Reasoning*, In Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA, pp. 329-334, August 15-17.

[Roychoudary et al., 2006] I. Roychoudhury, G. Biswas, and X. Koutsoukos, *A Bayesian approach to efficient diagnosis of incipient faults*, in Proc. 17th Int.Workshop Principles of Diagnosis, Jun. 2006, pp. 243–250.

[Chavira and Darwiche, 2007] M. Chavira and A. Darwiche. *Compiling Bayesian Networks Using Variable Elimination*, In Proc. of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07), January 2007, pp. 2443 – 2449.